GOAL User Manual

Koen V. Hindriks and Wouter Pasman

September 5, 2011

Contents

Chapter 1

Installing the GOAL Environment

The agent programming language GOAL is distributed with an Integrated Development Environment (IDE). This development environment provides a user interface that supports editing source code files, running a multi-agent system, debugging, and many more tasks needed for developing a GOAL multi-agent system. The environment thus facilitates development and running of GOAL agents that are part of a GOAL multi-agent system.

This document is the GOAL User Manual. It describes and explains how to use GOAL's development environment. It does not describe nor explain the agent programming language GOAL in any detail. For more information about the programming language and its features, we refer the reader to the GOAL Programming Guide [5] and GOAL's website [4].

1.1 Installation

This section discusses how to install and run the GOAL IDE. We briefly describe system requirements, the installation procedure, and how to start the GOAL IDE.

1.1.1 System Requirements

To install and run GOAL, you need

- a computer with the Windows, Macintosh OSX or Linux operating system. Precise requirements change over time and are detailed in the installation section on the GOAL website `http://mmi.tudelft.nl/trac/goal` where you can also download the latest version of the GOAL installer.
- SUN Java, currently version 1.6 or higher.

Note We have successfully run GOAL on 64 bits Windows machines (running Windows 7). However, 64 bit Java is not supported at this moment because of issues with running SWI Prolog on such machines. In order to run GOAL on 64 bits Windows, you need to install the 32 bit JVM (download from the Sun website and make sure that the 32 bit JVM is the first one in the PATH environment variable).

1.1.2 Installation

In order to install GOAL, you need to run the GOAL installer. This installer can be used for Windows, Mac OSX, and Linux. To obtain the installer, download the GOAL installer from `http://mmi.tudelft.nl/trac/goal`.

Before installing GOAL, you need to first make sure that the right version of Java has been installed. See above for the Java version that needs to be available on on your machine. We recommend to use Sun Java.[1]

Run the installer and follow the instructions. It is recommended to install all files including e.g. SWI Prolog by the installer. The latter is advised because GOAL requires the right version of SWI Prolog and will not run with other versions.[2]

Note Agent environments (e.g. simulators, etc) that have been downloaded separately and your own GOAL projects can be saved wherever you think it is convenient. It may be a good idea, however, to install these outside the GOAL directory. This will allow for easy upgrading of the GOAL IDE without the risk of overwriting your own projects. You need to specify your preferred folder for your own environments folder and folder to store their GOAL agents. To do this, use the Platformmanager and Environments tabs in the Help/Preferences menu.

Installation for Admins

GOAL can be installed in the `Program Files` directory (or another shared directory on OSX or Linux) so that multiple users can share the same installation.

The GOAL installer is not fully prepared for installation on Windows 7 in the `Program Files` directory (or any other directory with restricted access) because of some limitations of the installer we use. To install in such a directory, you need to log in as administrator or manually switch to administrator mode. On Windows 7, this works as follows:

1. right click on start menu / all programs / Accessories / Command prompt and select "Run as administrator"
2. Select "Yes" as windows asks you "Do you want to allow the following program to make changes to this computer".
3. cd <directory where you downloaded the `goalxxx.jar` installer> .
4. java -jar goalxxx.jar
5. follow the steps for installation. Select the required protected directory as install directory, e.g. `C:\Program Files\GOAL`
6. you can make shortcuts for all users (default on).

1.1.3 Uninstall and Update Versions

In order to uninstall GOAL, run the uninstaller that can be located in the main `GOAL` folder in which GOAL was installed in the uninstaller folder. Note that running the uninstaller will remove *all* installed files, possibly including any examples and environments that you modified!

Warning If GOAL was installed using an administrator account, it needs to be de-installed using the same administrator account. If you try to de-install without administrator rights the de-installer will fail without any error message.

To update GOAL, first run the uninstaller, and then repeat the usual installation procedure above. Alternatively, simply run the GOAL installer and say yes when you are asked whether you want to overwrite old files.

[1] We have found that some non-Sun Java fail to run the installer.

[2] The installer will automatically detect whether the JVM you use is 32 or 64 bits and will install the corresponding version of SWI Prolog. Make sure that you have SWI Prolog version 5.8.3 installed including jpl [7], if you want to use your own SWI Prolog installation. Also note that existing SWI Prolog installations on your machine will not be affected and can be run without problems after installing GOAL.

Chapter 2

Creating and Modifying Multi-Agent Systems Projects

The Integrated Development Environment (IDE) distributed with GOAL allows you to create new, and to load and modify existing GOAL multi-agent system projects. This chapter explains how to use the IDE to manage a multi-agent project.

2.1 Starting the GOAL IDE

To start the GOAL IDE, you can either double-click on the GOAL icon (if it has been created during installation), or selecting GOAL from the program menu.[1] After opening, the IDE window appears as illustrated in Figure 2.1. The GOAL IDE starts in *edit mode*.

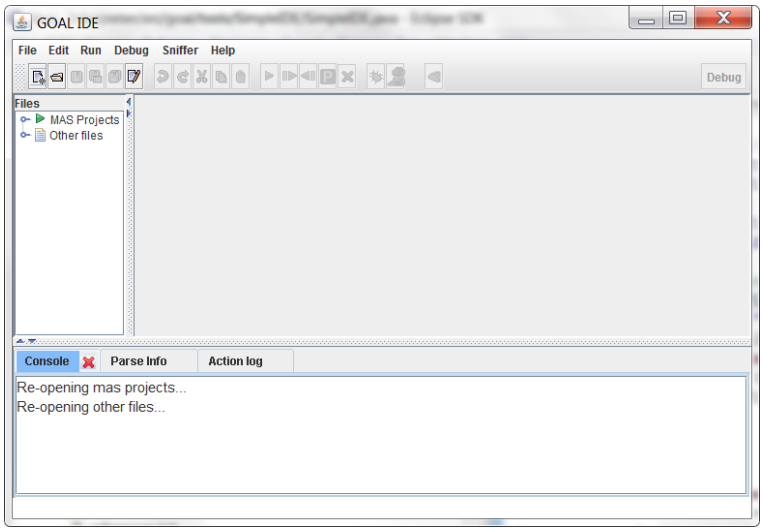

Figure 2.1: The GOAL IDE window right after start-up.

The main IDE window consists of four parts:

1. At the top of the window, the *menu bar* is located. The menu provides various functions for file management, editing mas projects, and running mas projects. Most menu items have

[1] Alternatively, it is possible to start the GOAL IDE via executing the `goal.exe` file (Windows), `goal.app` file (OSX), or the `goal` file (Linux) in the main `GOAL` folder. Finally, the IDE can also be started by executing the `goal.bat` (Windows) in the same folder, or executing `goal.sh` on the command line (Linux, OSX).

Chapter 4

Creating Your Own Environments

4.1 Locating Pre-Installed Environments

GOAL is distributed with several environments for which you can write your own agents. These environments are included in the `environments` folder in the main `GOAL` folder in which GOAL is installed. Running these environments is explained in Chapter 3. For ease of reference, you should add environment files that you create yourself to the `environments` folder as well. Alternatively, you can use your own folder and either specify references to this folder in the environments section of your mas files or change the reference to the folder that contains your environments via the preference panel shown in Figure 4.1.

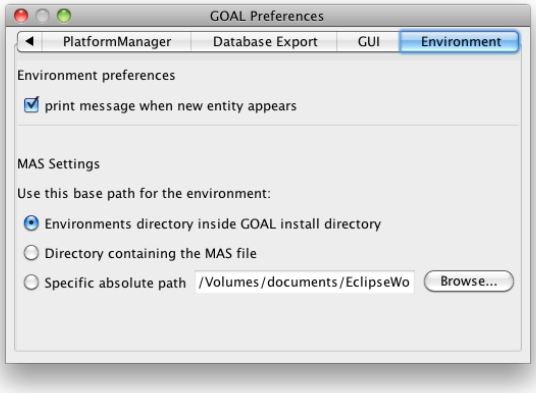

Figure 4.1: Preferences related to Environments

4.2 Creating an Environment Interface

Environments are viewed as a resource that provides so-called *controllable entities* which can be instructed by GOAL agents. More on this can be found in the GOAL Programming Guide [5]. An interface is used to connect GOAL agents to such entities. The interface used by GOAL is the Environment Interface Standard (EIS; [2, 1]). GOAL is fully compliant with EIS v0.3.

EIS is an Open Source project and documentation on how to create an interface for your environment such that it can operate in combination with GOAL is provided on the website [1]. Additionally, we have created some guidelines for how to create an environment on GOAL's webpage

[4] on the wiki containing Developer's Resources. Please mail to `goal@mmi.tudelft.nl` if you want to get access to these guidelines.

Chapter 5

Running GOAL in Stand-alone Mode

You can run a multi-agent system containing GOAL agent programs stand-alone, that is without starting the IDE. That way you can use GOAL from scripting languages like a shell script or bat script, or from your Java program.

Stand-alone running always involves starting up a Java Virtual Machine (JVM) with the right environment settings. We first describe the required environment and then describe how to use this from a Java program or from a script.

5.1 Setting the JVM environment

The variables to be set up properly for the environment are the `PATH`, the `JAVA_LIBRARY_PATH`, `CLASSPATH` and the `SWI_HOME_DIR`. Below we use the tag `GOALHOME` to point to the absolute directory where you installed GOAL. The exact way to do this is machine dependent; we assume you have the knowledge how to do this on your particular machine. For example, the ';' used as separator in the `CLASSPATH` below should be replaced with a ':' on Linux and OSX.

- on Windows, `PATH` should be set to include `GOALHOME/swifiles/libs`. On Linux you should set `LD_LIBRARY_PATH` instead, and on OSX you should set `DYLD_LIBRARY_PATH`.
- `SWI_HOME_DIR` should be set to point to `GOALHOME/swifiles/`.
- `JAVA_LIBRARY_PATH` should be set to include `GOALHOME/swifiles/libs`.
- `CLASSPATH` should be set to include `GOALHOME/goal.jar` and to the (EIS)environment jar file that you want to use.

You can inspect the goal.bat (Windows) or goal.sh file (Linux, OSX) to see a running example the above.

5.2 from Java

Running a stand-alone mas from Java is very simple: just create the goal.tools.Standalone object from java using this call

```
new goal.tools.Standalone(fileName);
```

where filename is an absolute path to the mas file you want to run.

5.3 from the Shell

To run GOAL stand-alone from the shell, the typical call looks like this:

```
java [-d32] -cp goal.jar -Djava.library.path=swifiles/libs goal.tools.Standalone
<filename.mas2g>
```

where `<filename.mas2g>` is the absolute path to the MAS file you want to run. The `-d32` option is used only when you are running a JVM that supports both 32 and 64 bit architecture and you have the 32 bit libraries for SWI prolog installed.

5.4 Runtime Settings

Some settings, e.g. used middleware, are used as set in the preferences. Note that these preferences can be set using the GOAL IDE, but also by direct calls.

Please refer to the JavaDoc for the various preference settings inside the `goal.tools.SimpleIDE.preferences` directory. There are preference settings for (amongst others):

- the PlatformManager, in PMPrefPanel, e.g. the search directory for environments, the location for the RMI registry (relevant if you use RMI middleware),
- the Warning setting in WarningPrefPanel, determining how many times errors are shown
- IDEPrefPanel, a.o. determining where messages are written to. If you want to see debug output in the stand-alone mode, you should turn off the Preferences/GUI/Show console output in IDE's console.

5.5 Interaction with Standalone

In many cases it is unavoidable to have user interaction in a stand-alone run. This is because the environment needs additional setup actions before it can really be used. Generally the agents appear in GOAL only after these additional actions have been taken.

Therefore, the Standalone waits for you to press the key 'r' (and the return key). Then it starts all the agents that are available in the middleware at that moment. Agents entering later will be put to pause mode (which is the initial run mode in GOAL at this time).

The standalone runner will check the environment run mode after the key press. If the environment is in paused mode, it will start the environment.

Finally, most mas do not have a natural end point. Therefore standalone kills a mas when you press the key 'x' (and the return key).

Chapter 6

Known Issues and Work Arounds

Below we have listed some typical problems that you may encounter. These problems typically occur due to either incorrect installation, syntax errors in the GOAL program, or represent some known issues.

- Fixing a Network issue (Windows Only) On Windows, there is a problem if you want to run the GOAL system with JADE, without having a network available. The problem is caused by the JADE subsystem that needs an IP address to run, and without network the computer has no IP address on Windows.

  A workaround for this issue is to use the Windows Loopback adapter. To install the adapter, do the following steps. You need administrator rights to do this.

  - go to "add hardware" in the control panel
  - select "yes, I have already connected..."
  - select "add new hardware device"
  - "Install the hardware that I manually select from a list"
  - "network adapters"
  - select Microsoft and then select "Microsoft Loopback Adapter"
  - finish the installation
  - go to Control Panel / Network Connections
  - select an ip address for the Loopback adapter. The number seems not really relevant, we tried 111.0.0.1 with success.
  - Check that the status indicated in the Network Connections window for the loopback adapter is "Limited or no connectivity".

- `Exception in thread "...." java.lang.UnsatisfiedLinkError:...` This most likely indicates that a dynamically linked library as installed by SWI Prolog could not be found. Did you install SWI Prolog using the GOALIDE installer? If not, your current installation of SWI Prolog is probably incompatible, incomplete or not set up properly to work in combination with GOAL.

- The Step and Run buttons seem to do nothing.
  There are a number of possible explanations for this you need to check.

  1. First, make sure that your environment is ready to run. For example, when using the Elevator environment the environment will respond to GOALonly if you selected *GOAL Controller*.

2. You may be stepping an agent that is suspended somehow, e.g. the environment may be refusing to serve that agent, or it may be the turn of another agent. Typically, this is resolved by selecting the mas node in the Process Panel and then continue stepping.

- The system fails to start up JADE. You may get a `jade.core.ProfileException` or `java.netMalformedURLException`.

  1. An instance of JADE is already running. Terminate the JADE instance that is running and try to launch your mas project again.
  2. On Windows there is an issue when you do not have a network available. Check out Section 6 for a workaround.

- The environment appears not to work properly.
  Check the documentation that comes with the environment. Most environments need to be set up before the agents can enter and perform actions.

- In the Wumpus environment, the system seems to hang when the agent shoots.
  The Wumpus World always has to be enclosed by means of walls. If you shoot the arrow into open space, the arrow will continue forever, causing the system to hang.

- Reset seems not to work.
  Known issue. Please refer to 3.2.[**TODO:** CHECK] Also check the console, it may produce a warning.

- When a mas has been launched, the environment is not visible.
  The environment may be hidden behind the GOAL IDE. This is due to a known bug in Java for Windows. You need to install java 7 or higher to fix this issue. Java 7 is available from `http://download.java.net/jdk7/binaries/`.

- Occasionally there can appear a message `Warning: No mapping found for expired timer 1237484141952 .`
  You can safely ignore such messages. It appears when a JADE agent is at a single moment in time both receiving a message and being scheduled in for execution.

6.1 Reporting Issues

Before reporting any issues, make sure you have always checked the Feedback Area, in particular the Console and Parse Info tabs for problems. If you find any bugs or have any issues, please report them to `goal@mmi.tudelft.nl`. To facilitate a quick resolution, please provide us with the following when you send in an issue report:

1. All the multi-agent system project files.
2. The relevant Console output, and/or Parse Info output. Include any stack trace output as well, if available (see the 3.6.2 for adjusting settings to obtain stack trace output).
3. Possibly a screen shot to clarify the situation you ran into.

Bibliography

[1] Tristan Behrens. Environment interface standard project. `http://sourceforge.net/projects/apleis/`.

[2] Tristan Behrens, Koen V. Hindriks, and Jrgen Dix. Towards an environment interface standard for agent platforms. *Annals of Mathematics and Artificial Intelligence*, pages 1–35, 2010. 10.1007/s10472-010-9215-9.

[3] Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood, editors. *Developing Multi-Agent Systems with JADE*. Number 15 in Agent Technology. John Wiley & Sons, Ltd., 2007.

[4] Koen V. Hindriks. The goal agent programming language. `http://mmi.tudelft.nl/~koen/goal.php`, 2011.

[5] Koen V. Hindriks. *GOAL Programming Guide*. Delft University of Technology, 2011. `http://mmi.tudelft.nl/~koen/goal.php`.

[6] open source platform. Jade: Java agent development framework. `http://jade.tilab.com`, 2010.

[7] Jan Wielemaker Paul Singleton, Fred Dushin. Jpl: A bidirectional prolog/java interface. `http://www.swi-prolog.org/packages/jpl/`, 2004.

[8] Open Source Platform. jedit: Programmer's text editor. `http://www.jedit.org/`, 2010.